https://ntrs.nasa.gov/search.jsp?R=19830027672 2020-03-21T01:09:19+00:00Z

NASA CR-165,811

3 1176 00160 0395

NASA Contractor Report 165811

NASA-CR-165811
19830027672

PAN AIR GEOMETRY MANAGEMENT SYSTEM (PAGMS) - A DATA-BASE MANAGEMENT SYSTEM FOR PAN AIR TYPE GEOMETRY DATA

Jon F. Hall

KENTRON INTERNATIONAL, INC.
Hampton Technical Center
an LTV Company
Hampton, Virginia 23666

Contract NAS1-16000
November 1981

~~FOR NASA AND NASA CONTRACTORS ONLY~~
Limitation removed per Ruth L. Tell/Fac. DAA Rec'd and dtd 9-20-83 S/Wallace Signed ma Mc 5-5-87

National Aeronautics and Space Administration

Langley Research Center
Hampton, Virginia 23665

NF01336

CONTENTS

N83-35943#
~~N82-10086~~ #

PAN AIR GEOMETRY MANAGEMENT SYSTEMS (PAGMS) - A DATA-BASE MANAGEMENT SYSTEM FOR PAN AIR TYPE GEOMETRY DATA

by

Jon F. Hall

Kentron International, Inc.

Hampton, Virginia 23666

SUMMARY

A data-base management system called PAGMS (Pan Air Geometry Management System) has been developed to facilitate the data transfers in applications computer programs that create, modify, plot or otherwise manipulate PAN AIR type geometry data in preparation for input to the PAN AIR system of computer programs. PAGMS is composed of a series of FORTRAN callable subroutines which can be accessed directly from applications programs. Currently only a NOS version of PAGMS has been developed.

INTRODUCTION

PAN AIR (reference 1) is a system of computer programs which perform linear theory subsonic or supersonic analyses of complex aircraft configurations. Use of PAN AIR requires that the input of the configuration surface geometry be divided into networks of panels defined by X, Y, Z Cartesian coordinates. A network can be thought of as a patch of the configuration surface with some logical significance such as the upper surface of a wing or the starboard side of a fuselage. Networks are further subdivided into quadrilateral panels which are defined by the coordinates of four corner points as described in the PAN AIR User's Manual (reference 2).

Creation of this often extensive geometry data (e.g. by digitizer), its intermediate manipulation (e.g. by computer graphics), and its subsequent output in a form compatible with the PAN AIR system require a software tool to efficiently handle the resulting large and often numerous data transfers between disk and core memory by applications type computer programs. A data-base management system called PAGMS (Pan Air Geometry Management System) has been developed to facilitate the manipulation of network geometry data.[1] The purpose of this report is to describe the structure and use of PAGMS.

---

[1] PAN AIR has its own data-base management system (SDMS) to handle geometry data once it has been received. The two systems are distinct and should not be confused.

DISCUSSION

The major impetus for developing PAGMS was to provide fast random access to network data from interactive graphics programs. As high speed and short response times are particularly desirable characteristics in interactive computer programs, simplified bulk data storage and retrieval becomes a necessity. This is especially true when large numbers of data transfers are involved. Achieving this performance requires the ability to address disk memory directly. As Langley Research Center supports both the NOS and PRIME computer systems, random disk addressing can be accomplished through the use of CDC Record Manager routines on NOS and their counterpart file manipulation routines on PRIME.

For purposes of program portability and NOS to PRIME compatability, programs should be as free of machine dependent code as is practical. While this may not be completely achievable, an effort is being made to group machine dependent code into well documented modules. Version 1.0 of the PAN AIR Geometry Management System is currently available on NOS as a library of FORTRAN callable routines that will have identical arguments with the PRIME version under development.[2] Both libraries will be compiled with their respective FTN compilers. These subroutine calls can be included in applications programs to efficiently handle network data manipulations.

---

[2] A PRIME version is currently being developed and will be released at a later date.

THE PAGMS LIBRARY

The PAGMS library of subroutines was designed to retrieve, store and otherwise manipulate a fundamental unit of data called a network. The PAGMS network consists of three one-dimensional arrays of specified length containing X, Y, Z Cartesian coordinate data. Each netweork is uniquely identified by the user and to the system through the use of a network identifier consisting of up to twenty alphanumeric characters. All references to network data must include the network identifier.

To use PAGMS, the data base must first be opened by calling routine OPENDB with a data base name, the name of the disk file on which the data base is to reside, and the TAPE/UNIT number to which error messages are to be written. To create networks in the data base from collected sets of data, subroutine ADDNET is called with a specified network identifier. In a similar fashion GETNET retrieves a network, DELNET deletes a network, REPNET replaces a network and RENAME renames a network. Calling subroutine CATLOG will produce a catalog listing of all network information on the data base. To make all data base transactions permanent, CLOSDB is called to close the data base.

A description of the function of each subroutine and its argument list is found in appendix A. The data base file structure, organization and implementation for the NOS version can be found in appendix B.

CONCLUDING REMARKS

A data-base management system has been developed to facilitate the data transfers in applications program which create, modify, plot, or otherwise manipulate PAN AIR type network geometry data. PAGMS (Pan Air Geometry

Management System) is a series of FORTRAN callable subroutines which may be accessed directly by applications programs executing on NOS. A PRIME version is under development. The system provides an efficient means for handling the often extensive data associated with PAN AIR networks for configuration modeling.

APPENDIX A

PAGMS SUBROUTINES

I. OPENDB

A. Function

This subroutine opens the data base, initializes the system common memory and examines a file status word to determine whether or not the file currently exists. If it does not, then a creation run is assumed and a file information table and a zero filled network directory are created and copied to the data base. If the file currently exists, its contents are copied to local scratch file SFLZZZ. All subsequent data base transactions will be performed on this scratch file until such time as the data base is closed and made permanent. This feature is used as a safeguard in the event of premature job termination due to system failure or user error. Note: Only one data base may be opened at any given time.

B. Calling Sequence

CALL OPENDB (MFILE, IDGEOM, MUNIT)

C. Arguments

1. MFILE - Input alphanumeric array. Specifies the name of the file (limited to 7 alphanumeric characters) on which the data base currently resides. If the file is being opened for the first time, this is the name of the file on which the data base is to be created. On re-opening a data base, it is the user's responsibility to ensure that MFILE is a local file, otherwise OPENDB will treat it as a creation run and open a new file.
2. IDGEOM - Input alphanumeric array. Contains 80 alphanumeric characters to be used at the discretion of the user to either

identify the configuration on the data base, name the data base or provide header information. IDGEOM is output when subroutine CATLOG is called. IDGEOM is a dummy variable on all but the first call to OPENDB.

3. MUNIT - Input variable, integer number. Specifies the number of the tape or unit to which all error messages are to be written. If MUNIT is set to zero, all error messages will be suppressed. Regardless of the input value of MUNIT, the NOS version places non-zero error codes in the user's dayfile.

II. CATLOG

A. Function

This subroutine catalogs the contents of the data base. It essentially dumps the contents of the file information table and the network directory to a specified tape unit. CATLOG provides the following information:

1. New or old data base.
2. Name of file on which data base resides.
3. 80 character data base identifier.
4. Creation date/time.
5. Last access date/time.
6. Network information.
   - 6.1 Network identifier (20 alphanumeric characters).
   - 6.2 Activity flag (active or inactive).
   - 6.3 Number of columns or N-Lines.
   - 6.4 Number of rows or M-Lines.
   - 6.5 Number of panels.
   - 6.6 Storage block type (an indication of the amount of storage reserved for the network).
7. Total number of networks (active and inactive).
8. Number of deleted or inactive networks.
9. Number of active networks.
10. Number of networks added since creation.
11. Total number of panels in all networks.
12. Length of data base in 60 bit words for NOS, 32 bit words for PRIME.
13. Length in words for each storage block type.

Table A-I presents an example CATLOG listing.

B. Calling Sequence

CALL CATLOG (LUNIT)

C. Arguments

1. LUNIT - Input variable, integer number. Specifies the number of the tape or unit to which catalog information will be written.

III. CATLST

A. Function

This subroutine produces a concise listing of all active and inactive networks on the data base. Inactive networks are prefixed with an asterisk. This routine was intended to be used as a quick reminder to the user of the contents of the data base. No other information is provided. Table A-II presents an example CATLST listing.

B. Calling Sequence

CALL CATLST (LUNIT)

C. Arguments

1. LUNIT - Input variable, integer number. Specifies the number of the tape or unit to which CATLST information will be written.

IV. ADDNET

A. Function

This subroutine adds a network to the data base. Networks are stored in blocks of memory of predefined size. There are five block sizes: 64, 128, 256, 512, and 1024 words containing floating point data. Once the length of the network is computed from the product N x M, a block size is determined. One block is reserved for each X, Y, Z array. If the data base is being created for the first time, slightly more space is reserved for each network in anticipation of future network expansions. Currently the product of N and M cannot exceed 1024 (approximately 960 panels per network). Some data may be truncated and an error message printed if more than 960 panels are specified for a network.

If the data base has been previously created and is currently being updated, ADDNET will search for blocks presently being occupied by deleted networks and overwrite them to conserve file space. At this point, the history of the deleted network disappears. Otherwise ADDNET will append the new network to the end of the file.

All networks are stored in the N-line order (columnwise). When adding a network that is M-line ordered, ADDNET first transposes the network to N-line order before placing it in the database.

B. Calling Sequence

```
CALL ADDNET (NETID, N, M, X, Y, Z, IORDER, IERR)
```

C. Arguments

1. NETID - Input alphanumeric array. Contains 20 alphanumeric characters of network identification.

2. N - Input/Output variable, type integer. On input N contains the number of columns of N-lines in the network data set. If IERR=1, N becomes an output variable containing an adjusted value, $2 \leq N \leq 512$. See table A-II for additional details.
3. M - Input/Output variable, type integer. On input M contains the number of rows or M-lines in the network data set. If IERR=2, M becomes an output variable containing an adjusted value, $2 \leq M \leq 512$. See table A-II for additional details.
4. X, Y, Z - Input real arrays. Contain one-dimensional network coordinate data with length N x M.
5. IORDER - Input variable, type integer. Specifies the order in which the data is stored in X, Y and Z, i.e. by columns or by rows.

   For IORDER=1, if X, Y, Z are thought of as two-dimensional arrays, they would be dimensioned X(M, N), Y(M, N), Z(M, N) exactly. This is termed N-line order.

   For IORDER=2, if X, Y, Z are thought of as two-dimensional arrays, they would be dimensioned X(N, M), Y(N, M), Z(N, M) exactly. This is termed M-line order.
6. IERR - Output integer variable. On output, IERR returns an error code describing the success or failure of the data base request. See table A-II for the error codes.

V. GETNET

A. Function

This subroutine gets a network from the data base. It has a limited error recovery capability. In the event GETNET is requested to retrieve a previously deleted network, it may do so provided only one deleted copy exists. If successful, the network is completely reactivated. No guarantee is made however, since a previous ADDNET may have overwritten the network in question. X, Y and Z must be dimensioned at least 1024 in the calling routine.

B. Calling Sequence

CALL GETNET (NETID, N, M, X, Y, Z, IORDER, IERR)

C. Arguments

1. NETID - Input alphanumeric array previously defined.
2. N - Output integer variable previously defined.
3. M - Output integer variable previously defined.
4. X, Y, Z - Output real arrays previously defined.
5. IORDER - Input integer variable previously defined.
6. IERR - Output integer variable previously defined.

VI. DELNET

A. Function

This subroutine deletes a network from the data base. It does so in a pseudo sense by turning on the network delete bit. Essentially this means that the network in question is a candidate for being overwritten at the earliest opportunity, but can be reactivated prior to being overwritten using the GETNET command.

B. Calling Sequence

```
CALL DELNET (NETID, IERR)
```

C. Arguments

1. NETID - Input alphanumeric array previously defined.
2. IERR - Output integer variable previously defined.

VII. REPNET

A. Function

This subroutine replaces the old network with a new one having the same NETID. Essentially old X, Y, Z data is overwritten by new data unless the new network requires a larger block size, in which case REPNET automatically deletes the old network and adds a new one.

B. Calling Sequence

CALL REPNET (NETID, N, M, X, Y, Z, IORDER, IERR)

C. Arguments

1. NETID - Input alphanumeric array previously defined.
2. N - Input integer variable previously defined.
3. M - Input integer variable previously defined.
4. X, Y, Z - Input real arrays previously defined.
5. IORDER - Input integer variable previously defined.
6. IERR - Output integer variable previously defined.

VII. RENAME

A. Function

This subroutine searches the data base for the first occurrence of network identifier NAMOLD and renames it NAMNEW irrespective of the delete bit status.

B. Calling Sequence

CALL RENAME (NAMOLD, NAMNEW, IERR)

C. Arguments

1. NAMOLD - Input alphanumeric array. Specifies the old 20 character network identifier that is to be replaced.
2. NAMNEW - Input alphanumeric array. Specifies the new 20 character network identifier that is to be associated with the old network data.
3. IERR - Output integer variable. On output IERR returns an error code describing the success or failure of the data base request. See table A-III for error codes.

IX. GETNAM

A. Function

This subroutine returns the names of all "active" network identifiers and the total number of them. The user should be aware that the data base can contain at most 100 networks (active plus inactive) and must dimension his arguments in the calling routine appropriately.

B. Calling Sequence

```
CALL GETNAM (NETIDS, NUMNET)
```

C. Arguments

1. NETIDS - Output alphanumeric 2-D array. Specifies the 20 character network identifiers. It is recommended that NETIDS be dimensioned: NETIDS (2,100) - NOS, NETIDS *4 (5,100) - PRIME.
2. NUMNET - Output integer number. Specifies the total number of active network identifiers returned in array NETIDS.

   $0 \leq \text{NUMNET} \leq 100$

X. CLOSDB

A. Function

This subroutine closes the data base making permanent all data base transactions. On creation runs, the master file is closed. On modification runs all "active" networks on the scratch file are copied to the master file thereby permanently eliminating all remaining deleted networks. The scratch file SFLZZZ is returned and the updated master file is closed. File MFILE is not replaced and remains a local file after closing.

B. Calling Sequence

CALL CLOSDB

C. Arguments

None.

TABLE A-I - EXAMPLE OF SUBROUTINE CATLOG OUTPUT

CATALOG OF PANEL DATA ON OLD FILE GET25

TWIN BODY DATABASE FROM HESS GEOMETRY GETR25

CREATION DATE/TIME 81/10/21. 11.21.06.
LAST ACCESS DATE/TIME 81/10/21. 11.27.06.

NETWORK IDENTIFIER 20 CHARACTERS	ACTIVITY FLAG	COLUMNS N-LINES	ROWS M-LINES	NUMBER OF PANELS	BLOCK TYPE
NOLIFT_SRF001	ACTIVE	2	14	13	1
NOLIFT_SRF002	ACTIVE	2	14	13	1
NOLIFT_SRF003	ACTIVE	2	14	13	1
NOLIFT_SRF004	ACTIVE	2	14	13	1
NOLIFT_SRF005	ACTIVE	9	11	80	3
NOLIFT_SRF006	ACTIVE	9	11	80	3
NOLIFT_SRF007	ACTIVE	14	11	130	3
NOLIFT_SRF008	ACTIVE	14	11	130	3
NOLIFT_SRF009	ACTIVE	9	11	80	3
NOLIFT_SRF010	ACTIVE	9	11	80	3
NOLIFT_SRF011	ACTIVE	2	11	10	1
NOLIFT_SRF012	ACTIVE	2	11	10	1
NOLIFT_SRF013	ACTIVE	2	11	10	1
NOLIFT_SRF014	ACTIVE	2	11	10	1
NOLIFT_SRF015	ACTIVE	14	8	91	3
NOLIFT_SRF016	ACTIVE	14	8	91	3
NOLIFT_SRF017	ACTIVE	2	14	13	1
NOLIFT_SRF018	ACTIVE	2	14	13	1
TEMPNET297 13.42.43.	*DELETED*	2	11	10	1
TEMPNET786 13.42.32.	*DELETED*	9	11	80	2

TOTAL NUMBER OF NETWORKS	20
NUMBER OF DELETED NETWORKS	2
NUMBER OF ACTIVE NETWORKS	18
NETWORKS ADDED SINCE CREATION	2
TOTAL NUMBER OF PANELS	970
LENGTH OF DATABASE (WORDS)	9088
LENGTH OF BLOCK TYPE 1	64
LENGTH OF BLOCK TYPE 2	128
LENGTH OF BLOCK TYPE 3	256
LENGTH OF BLOCK TYPE 4	512
LENGTH OF BLOCK TYPE 5	1024

TABLE A-II - EXAMPLE OF SUBROUTINE CATLST OUTPUT

```
 NOLIFT_SRF001          NOLIFT_SRF002          NOLIFT_SRF003
 NOLIFT_SRF004          NOLIFT_SRF005          NOLIFT_SRF006
 NOLIFT_SRF007          NOLIFT_SRF008          NOLIFT_SRF009
 NOLIFT_SRF010          NOLIFT_SRF011          NOLIFT_SRF012
 NOLIFT_SRF013          NOLIFT_SRF014          NOLIFT_SRF015
 NOLIFT_SRF016          NOLIFT_SRF017          NOLIFT_SRF018
*TEMPNET297 13.42.43. *TEMPNET786 13.42.32.
```

TABLE A-III PAGMS ERROR CODES

IERR	MEANING	SUGGESTED USER CORRECTIVE ACTION
0	No Error.	None
1	Attempt to add N-Line ordered network longer than maximum allowed block size. Some N-Lines were deleted so that network could be saved.	Reduce network size to prevent truncation.
2	Attempt to add M-Line ordered network longer than maximum allowed block size. Some M-lines were deleted so that network could be saved.	Reduce network size to prevent truncation.
3	Attempt to add N-Line ordered network longer than maximum allowed block size. M must be $\leq$ 512. No action taken.	Reduce M.
4	Attempt to add M-Line ordered network longer than maximum allowed block size. N must be $\leq$ 512. No action taken.	Reduce N.
5	Too few M-lines. M must be $\geq$ 2. No action taken.	Increase M.
6	Too few N-lines. N must be $\geq$ 2. No action taken.	Increase N.
7	Network to be added already permanent. No action taken.	Replace network if old data not important, or add network with new name.
8	Data base overflow, number of active and inactive networks exceeds 100. No action taken.	Close data base and re-open.
9	Network not found under specified identifier. No action taken.	Check spelling.

CONTINUED ⟶

TABLE A-III PAGMS ERROR CODES

CONTINUED

IERR	MEANING	SUGGESTED USER CORRECTIVE ACTION
10	Attempt to get network that has been previously deleted. Duplicate copies exist making recovery impossible. No action taken.	Attempt to "RENAME" first deleted copy.
11	Attempt to get network that has been previously deleted. Single copy exists, network reactivated.	None needed.
12	Attempt to rename a network with a non-unique identifier. No action taken.	Use new name.
13	Network directory empty. Attempt to get network from empty data base. No action taken.	Create data base by adding networks.
14	Network directory empty. Attempt to delete/replace network from empty data base. REPNET will revert to ADDNET and add the network.	Create data base by adding networks.
15	Attempt to rename a network from an empty data base. No action taken.	Create data base by adding networks.

APPENDIX B

PAGMS FILE STRUCTURE - NOS VERSION

DATA ACCESS METHOD

The NOS version of PAGMS uses the Control Data Corporation Cyber Record Manager directly. (Detailed information about Record Manager can be found in CDC publication 60495700 entitled "Cyber Record Manager, Basic Access Methods, Version 1.5 Reference Manual"). The advantages over the FORTRAN IV extended READMS/WRITMS routines are threefold. First, when I/O logical records can be restricted to multiples of PRU's (physical record units), the filed length I/O buffer set up by the compiler can be suppressed. This results in reduced field length and faster internal data transfer, both important considerations in interactive programs. Secondly, a natural key (index pointer) to network data is the 20 character network identifier that PAN AIR uses to label each network. On NOS this identifier occupies two words and is thus unacceptable as a key to READMS/WRITMS since named indexing only accomodates one word of 10 characters as a key. Lastly, because of limits placed by PAN AIR on the number and sizes of networks, data base bookkeeping is greatly simplified making it less of a nuisance to the programmer who has to perform it. PAN AIR restricts an input configuration to a maximum of 100 networks. The total number of panels per configuration has a practical limit of 1200-1500 due to file storage and execution costs.

To this end, all data is stored on WA (word addressable) type files with U(undefined) type logical records. Buffering is suppressed.

FILE INFORMATION TABLE

The PAGMS file structure consists of several logical records described as follows. The first is called the File Information Table and has a fixed length of 48 words. This table includes such items as the 80 character name of the data base, the date and time of its creation and last access, the total number of networks, the number of entries in the Network Directory (yet to be defined) and the address of the first word past the end of file. As shown in Table B-1, free space has also been included for additional information.

NETWORK DIRECTORY

The next logical record is called the Network Directory and has a fixed length of 400 words. The directory is subdivided into 100 groups of 4 words each. Each group identifies and points to a particular network data set. The first two words of every group contain the 20 character network identification. The third word contains the address of the first word (an x-coordinate value) of the corresponding network data set. Packed into the fourth word are the delete bit, the block type and the values of N and M (the network length). Table B-II illustrates the Directory structure. During processing, copies of both the File Information Table and the Network Directory are kept in core to gain speed in addressing network data sets.

NETWORK DATA SETS

Following the Network Directory are the Network Data Sets. Their record lengths vary with block type. There are five block types ranging

in length from 64 words to 1024 words. As shows in table B-III, each network data set consists of 3 records, one for each X, Y, and Z coordinate array.

EXECUTION CONTROL CARDS

The PAGMS library of routines (NOS version) is on an indirect access file named PAGMSLB which may be accessed by using a GET command:

```
GET, PAGMSLB/UN = 359950C.
```

To execute a program with this library a loadset card is needed. The following is an example when no other libraries are being loaded.

```
LDSET, LIB = PAGMSLB, PRESETA = INDEF.
LGO.
```

TABLE B-I - FILE INFORMATION TABLE

WORD	CONTENTS
1	File Status Word
2	First Word Address Past EOF
3	Data of Creation
4	Time of Creation
5	Date of Last Access
6	Time of Last Access
7	Total Number of Networks
8	Networks Added Since Creation
9	Networks Deleted Since Creation
10	Length of Directory
11	New/Old Flag
12	Data Base Header
13	Data Base Header
14	Data Base Header
15	Data Base Header
16	Data Base Header
17	Data Base Header
18	Data Base Header
19	Data Base Header
20	Block Size 1
21	Block Size 2

CONTINUED →

TABLE B-I - FILE INFORMATION TABLE CONTINUED

WORD	CONTENTS
22	Block Size 3
23	Block Size 4
24	Block Size 5
25	XXXXXXX
.	.
.	.
.	.
48	XXXXXXX

XXXXXXX - ZERO FILL

TABLE B-II - NETWORK DIRECTORY

WORD	CONTENTS				
	59	58 - 56	55 - 46	44 - 45	44 - 0
1	NETWORK IDENTIFIER WORD 1				
2	NETWORK IDENTIFIER WORD 2				
3	NETWORK KEY (ADDRESS POINTER)				
4	DEL	BLOCK TYPE	N	M	XXXXXXX
5	NETWORK IDENTIFIER WORD 1				
6	NETWORK IDENTIFIER WORD 2				
7	NETWORK KEY (ADDRESS POINTER)				
8	DEL	BLOCK TYPE	N	M	XXXXXXX
9	NETWORK IDENTIFIER WORD 1				
10	NETWORK IDENTIFIER WORD 2				
11	NETWORK KEY (ADDRESS POINTER)				
12	DEL	BLOCK TYPE	N	M	XXXXXXX
•					
•					
•					
•					
•					
397	NETWORK IDENTIFIER WORD 1				
398	NETWORK IDENTIFIER WORD 2				
399	NETWORK KEY (ADDRESS POINTER)				
400	DEL	BLOCK TYPE	N	M	XXXXXXX

XXXXXXX - ZERO FILL

TABLE B-III - NETWORK DATA SETS

1st Word Block Type I (1 - 5) Last Word	X (1) X (N * M) Free Space
1st Word Block Type I (1 - 5) Last Word	Y (1) Y (N * M) Free Space
1st Word Block Type I (1 - 5) Last Word	Z (1) Z (N * M) Free Space

REFERENCES

1. Carmichael, R. L. and Erickson, L. L.: PAN AIR - A Higher Order Panel Method for Predicting Subsonic or Supersonic Linear Potential Flows about Arbitrary Configurations. AIAA Paper 81-1225, June 1981.

2. Sidwell, Kenneth W.; Baruah, Pranab K.; and Bussoletti, John E.: PAN AIR - A Computer Program for Predicting Subsonic or Supersonic Linear Potential Flows about Arbitrary Configurations Using a Higher Order Panel Method. Vol. II - User's Manual (Version 1.0), NASA CR 3252, May 1980.

1 Report No	2 Government Accession No	3 Recipient's Catalog No
NASA CR-165811		

4 Title and Subtitle	5 Report Date
PAN AIR GEOMETRY MANAGEMENT SYSTEM (PAGMS) - A DATA-BASE MANAGEMENT SYSTEM FOR PAN AIR TYPE GEOMETRY DATA	November 1981
	6 Performing Organization Code
7 Author(s)	8 Performing Organization Report No
Jon F. Hall	
	10 Work Unit No
9 Performing Organization Name and Address	
KENTRON INTERNATIONAL, INC. Hampton Technical Center an LTV Company Hampton, Virginia 23666	11 Contract or Grant No NAS1-16000
	13 Type of Report and Period Covered
12 Sponsoring Agency Name and Address	Contractor Report
National Aeronautics and Space Administration Washington, DC 20546	14 Sponsoring Agency Code 505-43-23-02

15 Supplementary Notes

Langley Technical Monitor: David S. Miller

16 Abstract

A data-base management system called PAGMS (Pan Air Geometry Management System) has been developed to facilitate the data transfer in applications computer programs that create, modify, plot or otherwise manipulate PAN AIR type geometry data in preparation for input to the PAN AIR system of computer programs. PAGMS is composed of a series of FORTRAN callable subroutines which can be accessed directly from applictions programs. Currently only a NOS version of PAGMS has been developed.

17 Key Words (Suggested by Author(s))	18 Distribution Statement
PAN AIR, data base, geometry, data-base management	FOR NASA AND NASA CONTRACTORS ONLY

19 Security Classif (of this report)	20 Security Classif (of this page)	21 No of Pages	22 Price
Unclassified	Unclassified	32	

End of Document